ARI Contractor Report 98-08

Intelligent Dialog Tutor and Conversational Agents

Michelle Sams and William R. Murray
Teknowledge Corporation

William H. DeSmedt and Donald Loritz
Amber Consortium

July 1998

United States Army Research Institute for the Behavioral and Social Sciences

REPORT DOCUMENTATION PAGE

1. REPORT DATE (dd-mm-yy)	2. REPORT TYPE	3. DATES COVERED (from. . . to)
July 1998	Final Report	January 98 -June 98

4. TITLE AND SUBTITLE	5a. CONTRACT OR GRANT NUMBER
Intelligent Dialog Tutor and Conversational Agents	DASW001-98-M-0879
	5b. PROGRAM ELEMENT NUMBER 2Q665502

6. AUTHOR(S)	5c. PROJECT NUMBER
Michelle Sams and William R. Murray, Teknowledge Corporation	M770
	5d. TASK NUMBER
William H. DeSmedt and Donald Loritz. Amber Consortium	
	5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Teknowledge Corporation 4350 N. Fairfax Drive, Suite 420 Arlington, VA 22203	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. MONITOR ACRONYM
U.S. Army Research Institute for the Behavioral and Social Sciences Attn: TAPC-ARI-II 5001 Eisenhower Avenue Alexandria, VA 22333-5600	ARI
	11. MONITOR REPORT NUMBER Contractor Report 98-08

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES This report is published to meet legal and contractual standards and may not meet ARI's scientific or professional standards for publication.

14. ABSTRACT *(Maximum 200 words)*:

This report describes an approach to develop intelligent and realistic computer-based training environments that are easily authored by non-programmers. It focuses on simulated agents who are capable of mixed-initiative dialog with the user through a natural language interface, either via speech or text. Trainees converse as they would in normal English discourse; asking questions, giving answers, and making comments or requests. The simulated agent generates appropriate responses to all of these inputs within any defined domain. This dialog capability will be realized in a simulated tutor and in simulated characters for mission training scenarios. The Intelligent Dialog Tutor provides individualized instruction based on what the trainee knows and does not know. After the lesson, the trainee can practice the newly acquired skills in realistic scenarios populated with Conversational Agents. The system is fully authorable, thus the Tutor can instruct and discuss any topic, including conceptual (e.g., leader skills) and procedural domains (e.g., maintenance tasks). Conversational Agents are easily created for new scenarios. Technologies include natural language processing, semantic analysis, discourse management, agent control architectures, a generic tutoring shell with a knowledge elicitation tool, continuous speech recognition, text-to-speech synthesis, and distributed object-oriented software for optimal Internet delivery and management.

15. SUBJECT TERMS

intelligent tutoring system, computer-based training, natural language processing, intelligent agents, virtual environments, simulations, continuous speech recognition

SECURITY CLASSIFICATION OF			19. LIMITATION OF ABSTRACT	20. NUMBER OF PAGES	21. RESPONSIBLE PERSON (Name and Telephone Number)
16. REPORT	17. ABSTRACT	18. THIS PAGE			
Unclassified	Unclassified	Unclassified	Unlimited	39	Jonathan Kaplan (703) 617-8828

Intelligent Dialog Tutor and Conversational Agents

CONTENTS

INTRODUCTION 1
Army Need 1
Innovative Research 2

TECHNICAL OBJECTIVES AND APPROACH 3
Functional Overview 3
System and Components Overview 4

DETAILS OF THE TECHNICAL APPROACH 7
Intelligent Tutoring System 7
Integration of Tutor with Dialog System 10
Dialog Agent System 18
Spoken Interface and Talking Head 27
Overall System Control and Delivery 34

REFERENCES 35

LIST OF TABLES

Table 1. Product comparison table for spoken interface and talking head components. 33

LIST OF FIGURES

Figure 1. Authoring the Military Decision-making Process for the tutor Kb. CAT authoring screen at top automatically builds the model shown below. 14

Figure 2. Authoring Brigade Commander process to determine which staff officers to take to Division OPORD briefing. CAT decision authoring screen at top builds the model shown below. 15

Figure 3. Authoring an instructional unit in a lesson. 16

Figure 4. Sample Trainee Screen. Instructional material is presented first, then the Challenge. The trainee can ask or answer questions at any time. 17

Figure 5. A process diagram of the Dialog Agent System. 18

Figure 6. Overall system control and delivery. 34

INTRODUCTION

Army Need

The Army is undergoing major restructuring of its force and of its approach to meet their training needs. Training will be organized around real-time applications of information and information technology on the battlefield, in the units, and in the classroom to maintain a continuous edge in projecting and employing combat power. The proposed effort supports the Army's Warrior XXI objectives for computer-based training on stand-alone platforms or in distributed simulated environments to deliver training when and where needed.

The goal of this project is to develop simulated agents who are capable of mixed-initiative dialog with the user through a natural language interface, either via speech or text. Trainees converse as they would in normal English discourse; asking questions, giving answers, and making comments or requests. The simulated agent responds appropriately to all of these inputs within any defined domain. None of this discourse is canned, it is generated on the fly based on what the agent "knows" and what its goals are. For this project we will develop and apply this dialog capability in a simulated tutor and in simulated characters for mission training scenarios.

Most current computer-based training is electronic page turning, with some hyperlinks and multiple-choice exercises. While this electronic delivery provides self-paced learning, it does not adapt to the trainee and is not truly interactive. It is well documented that individualized instruction greatly enhances learning in a shorter period of time. However, developing a separate Intelligent Tutoring System (ITS) for every Army training topic is a prohibitively expensive undertaking. What is needed is a generic, easily authored ITS system. This, in fact, is exactly what we are prepared to build *plus* we will add an authorable natural language dialog capability.

The Intelligent Dialog Tutor will provide trainees with individualized instruction through natural language dialog with a simulated tutor agent. The tutor presents instruction and asks questions of the trainee. The trainee provides a freely formed response. The Tutor then evaluates the trainee's responses, gives specific feedback, and adapts the lesson based on what the trainee knows and does not know. Trainees can ask questions about the material at any time.

After the lesson, the trainee can practice the newly acquired skills in realistic scenarios populated with conversational agents. For example, the trainee could learn about the Military Decision-making Process with the simulated tutor and then practice these skills in a simulated Tactical Operation Center (TOC). The TOC would have simulated staff officers who can dialog about facts relevant to their positions and the battle scenario. We will provide the capability to create new simulated characters and new scenarios through easy authoring of their 'mindsets' (i.e., their knowledge bases).

As the DoD is moving towards more training in virtual simulations, there is a need to provide more realistic interactions with the simulated entities (e.g., tanks, dismounted troops). Currently these entities move about in the 2-D or 3-D representations according to pre-programmed rules of engagement, but do not communicate with the trainee. There are efforts currently underway to develop interfaces to enable simulated troops to recognize trainee gestures. However, there is no effort funded yet that can provide these simulated agents with the capability to carry on intelligent dialog with the trainees. For example, a dialog capability would allow mission rehearsals in which live commanders can ask simulated troops for information and give orders. The simulated troops can offer information and ask questions to clarify the

commander's intent. What we are talking about here is more than simple recognition of voice commands -- we are enabling the simulated agents to engage in realistic discourse. The dialog agent technology that we are developing in this Phase II effort can be leveraged to meet this need.

Our dialog agents will provide intelligent and realistic training environments that are easy to author by non-programmers. This meets the Army's need for just-in-time training for the rapidly changing missions and new battlefield technology.

Innovative Research

In ITS, as in related fields like knowledge representation, it is not at all unusual to find the door barred to "naive users" and "neophyte editors" (Lenat & Guha 1990, p. 30). The upshot has been to consign the design and development of tutoring systems to those whose credentials, while impressive, are in disciplines like "ontological engineering" or "computational linguistics" — all far removed from day-to-day instructional practice. Our project takes a different approach by devising an authoring interface for non-programmers for the ITS, conversational agents, and dialog system.

Our tutor can be authored for any domain, including conceptual (e.g., leader skills) and procedural domains (e.g., maintenance tasks). Most intelligent tutoring systems focus on expert models for specific precise, formal domains (e.g., avionics, electronic circuits). There have been a few authorable ITS, but none truly generic. The IMTS / RAPIDS / RIDES series of shells and toolkits focuses on simulation-based training with graphical device simulations. Earlier shells focused on exploring different research issues, (e.g., GUIDON, Clancey, 1987).

Our proposed tutorial system couples dynamic lesson planning with fully integrated natural language capabilities. Previous tutors such as the Lower Hoist Tutor (Murray, 1990) or Meno-Tutor (Woolf, 1984) may have simulated such interactions, but did not provide natural language input or generation without recourse to external systems. Our system will fully integrate lesson planning and discourse planning for the first time, resulting in more realistic tutorial dialog.

Thus, the innovative features that distinguish this proposal from other work are:

1. Intelligent tutoring system fully integrated with natural language capabilities, including mixed initiative discourse.
2. Role-playing with simulated conversational agents.
3. Fully authorable, domain independent.

TECHNICAL OBJECTIVES AND APPROACH

Functional Overview

First we will provide a brief, non-technical overview of the functionality of the Tutor, conversational agents, and the major system components. A more detailed specification of the technical approach follows.

Tutor Interaction

The trainee will be presented with instructional material (text, graphics, video, audio) and asked to answer a question or provide a narrative description. A response box is provided for the freely typed input. If the trainee decides to respond verbally, this response box also automatically displays the trainee's spoken input on the fly, as it is translated by the speech recognition system. If an error in speech recognition occurs, then the trainee can correct the response before it is submitted to the Tutor. The Tutor is represented by an animated talking head that is driven by a text-to-speech synthesizer.

After the trainee responds to the question, it is evaluated for accuracy and completeness. The Tutor then provides the appropriate response. A correct evaluation will trigger the Tutor to give affirmative feedback. A correct but incomplete response will trigger the Tutor to prompt the trainee for the specific missing information. An incorrect evaluation will trigger the Tutor to provide precise feedback and suggest that the trainee review specific underlying principles/steps. The Tutor then presents that relevant material.

The trainee has the option to ask the Tutor a question at any time while progressing through the lesson material. The trainee simply clicks on a box beneath the Tutor head, labeled "Talk to the Tutor". A pop-up dialog box occurs and the trainee can interact with the Tutor either by typed or spoken input. The Tutor evaluates the trainee query and responds appropriately, either answering the question or posing a question in return.

For example: The Tutor presents the staff responsibilities of the ADO, AVN LNO, and the AF LNO. Then the Tutor asks some questions to ascertain whether the trainee has acquired the knowledge.

Tutor: "Which staff officer templates air avenues of approach with the S2? "
Trainee: "The AF LNO."
Tutor: "That's not correct. The AF LNO provides input on enemy fixed-wing assets and high altitude ADA equipment. Would you like to suggest another officer?"
Trainee: "I am not sure whether it would be the ADO or the AVN LNO. "
Tutor: "Ok. Let's review. The AVN LNO provides information on enemy rotary-wing lift and attack assets and the ADO templates air avenues of approach. Do you understand the difference?"
Trainee: "Yes. I just forgot."
Tutor: "That's ok. You will remember more as we review and do some practice scenarios."

Conversational Agent Interaction

After the lesson, the trainee can practice the newly acquired skills in realistic scenarios populated with conversational agents. For example, the trainee learned about the military decision-making process with the simulated tutor. Then the Tutor suggests that the trainee select one of the skill practice scenarios in

the Tactical Operation Center (TOC). The trainee clicks on the button to start the scenario and is given a brief introductory narrative that describes the situation.

For example: *You are in the mythical country, Borania, on a contingency operation. There is guerrilla activity in the area, and there are some American citizens that need to be evacuated. You are the Executive Officer. Your brigade commander returns from division headquarters with the warning order to begin a deliberate attack in 72 hours. You are given brief and incomplete information about the enemy contact. Now go to the TOC and get the information you need to develop an operation order for this specific mission.*

The trainee clicks on the TOC button and sees a graphic representing his staff officers with a button under each one titled by position (e.g., S1 -personnel, S2 - intelligence). Each of these simulated staff officers is enabled by our conversational agent technology. Each has information about their respective domains and facts specific to the scenario. The trainee can ask them questions and make requests. The currently-active conversational agent can also initiate comments or questions (based on definable goals).

Trainee: "Attention in the TOC! Task Force Hawk has reported heavy contact on the high ground of South Objective ROOSTER."

Trainee: [Clicks on ALO button] ALO, what is the status on immediate air support?
ALO: "The 5th Squadron is en route to LZ 20, sir."
Trainee: "Any other air support?"
ALO: "The 7th could be a 'go' in 24 hours"
Trainee: "Send a WARNO to the 7th."
ALO: "Yes sir!"

Trainee: [Clicks on ADO button] "ADO, give me you best guess on enemy ADA threat."
ADO: "Sir, the enemy has a few STINGERs. Nothing else."

ADO: "Sir, a new report just in!"

System and Components Overview

Here we will give a brief non-technical overview of the components in our Dialog Agent System. Full technical descriptions follow. Our dialog system provides the capability for the agents to recognize, understand, and generate language -- "how to converse". Our knowledge representation system provides the agents with general concepts and specific facts -- "what to talk about". Our control strategies provide our agents with goals, reasoning and planning capability -- "how and when to say it".

Dialog Agent System (DAS)

The Dialog Agent System enables the simulated agents to 'understand' freely formed user input and to generate appropriate responses. When the user poses a question or makes a statement, the input is first parsed to identify what the words are and how they are related to each other (e.g., who did what to whom). The parser relies on several sub-components to do this job, including a morphological analyzer and a lexicon (similar to a dictionary).

The results of this syntactic analysis are enhanced with contextual information from the discourse management technology and mapped into a semantic case-frame to resolve discourse phenomena

(e.g., pronouns referring to previously mentioned people and objects). After the semantic and discourse analysis, the input is reduced to one or more propositions (or "factoids") expressed in a language-independent formalism.

The factoid is now passed to the knowledge manager. The knowledge manager's inferencing engine attempts to match the incoming factoid against the contents of the currently active knowledge base. Prior to formulating a candidate output response, the system checks whether anything in the user input or event state would cause the conversational initiative to pass from the agent to the user, or vice versa.

The dialog system now formulates a candidate response based on degree of resonance with the knowledge bases and the agent's goal. The candidate response factoid is then run back through a discourse-management filter to screen out elements that are not germane to the current discussion, and to accord appropriate emphasis to those which are. The resulting discourse-sensitive factoid is then fed into the language generator, which performs a mapping from semantic to syntactic case frames, applies surface transforms consistent with the focus of discourse, and formulates an output string for delivery to the learner.

Knowledge Representation for the Dialog Agent System (DAS)

The DAS knowledge-representation schema is a structured hierarchy of language-independent ideas, but is not a simple inheritance hierarchy. The schema can represent a wide range of factoids, from simple propositional relationships and declarative clauses to more complex relationships and concepts such as, deterministic and probabilistic logical and relational operators, implication, universal and existential quantification.

Intelligent Tutoring System

The Tutor has an expert model (a knowledge base) consisting of knowledge units (e.g., facts, terms, concepts, steps) and links that depict the relationships among those units. These links can indicate a number of different relationships, such as top level facts with underlying concepts or conditions that must be met before the next step is taken in a procedure. Instructional packets are associated with each of the knowledge units that require instruction. Instructional packets consist of material (e.g., narratives, graphics, audio) and Challenges (i.e., question-answer exercises).

The Tutor has an instructional planning strategy that evaluates the trainee's mastery of the topic and determines how best to achieve the lesson objectives for that individual. Progression through the knowledge space will be different for each trainee, depending on performance. A knowledgeable trainee will be moved rapidly through top-level nodes, perhaps occasionally dipping down to underlying nodes when the trainee demonstrates knowledge gaps. A novice will traverse more of the underlying nodes (e.g., terms, concepts, steps).

The Tutor employs a dynamic planning strategy to determine the nature of the tutorial dialog and the movement of the trainee through the network of domain knowledge. This planning strategy functions independently of the domain content. Movement through the knowledge space is determined by the types and strengths of relationships among nodes and the current state of the student model. Based on these factors, topics and exercise sequence are prioritized and an initial instructional plan prepared. This plan is continuously and dynamically updated/modified, not only based on performance but also driven by trainee questions and requests.

Spoken Interface and Talking Head

Users can choose either to type or speak their input. In turn, the system's output can be presented directly as text, or passed to a speech synthesizer accompanied by an animated talking head. As part of our Phase I effort, we reviewed several systems offering continuous speech recognition (CSR), text-to-speech synthesis (TTS), and talking heads. Evaluations were based on performance, interface and platform requirements, ease of use, technical support, and cost.

We concluded that IBM's *Via Voice* is the best overall package choice at this time. *Via Voice* is designed to be speaker independent and showed about a 96% speech recognition accuracy in our tests. The *Via Voice* package had the strongest TTS engine in the group we evaluated. It supplies eight pre-configured voices, as well as an easy authoring interface for creating a wide range of variable additional voices. The IBM developer kit also includes Virtual Voices, eight talking-head "actors" (man, woman, child, and 5 caricature actors) with six expressions that one can preset. The *Via Voice* package was easy to install and use, met our interface requirements, and in addition, IBM provided excellent technical support. *Via Voice* costs $100 per workstation.

Authoring Overview

To author the Intelligent Dialog Tutor, one first enters the tutor's domain knowledge via CAT ©(our template-driven knowledge elicitation tool). Next, lessons are authored with a simple template interface. This tutor and lesson output is automatically sent to the dialog system to check for words and concepts that are not represented. An Authoring Assistant agent engages in a dialog with the author to identify the meanings and relationships of any unknown words and concepts. The tutorial strategy is automatic, however authors can set some parameters via slider bars (e.g., degree of tutorial assistance such as hinting).

Authoring a conversational agent is easy. One simply names a character and then types in natural language the facts that he/she knows. Character personalities are modified via slider bars (e.g., from cooperative to reticent).

We are confident that we can provide an easy authoring system. We have already solved some of the challenges. CAT is operational, as well as the 'mindset' authoring for the conversational agents. We are currently developing lexicon and ontology authoring. Therefore, our main goal in this effort is to integrate the tutor and dialog authoring, and develop the slider bar authoring for the tutorial strategy and the character personality.

DETAILS OF THE TECHNICAL APPROACH

In the following sections we provide a more detailed explanation of our technical approach for the Intelligent Tutoring System, the Dialog Agent System, and the Spoken Interface and Talking Head.

Intelligent Tutoring System

Mixed Initiative Instruction

One of the key features of the proposed tutor is its support for mixed initiative instruction that mirrors the normal give and take of tutorial dialog. Previous tutors such as the Lower Hoist Tutor (Murray, 1990) or Meno-Tutor (Woolf, 1984) may have simulated such interactions but did not provide natural language capabilities themselves. For example, Meno-Tutor's output could be used as input to the MUMBLE (McDonald, 1983) natural language system but Meno-Tutor itself did not provide natural language output. Our proposed tutor will be directly integrated with the natural language understanding and generation capabilities of an authorable dialog system.

Mixed initiative instruction requires a flexible dynamic planning approach at multiple levels of plan abstraction. Having a plan, or being able to develop a plan once the trainee's needs are known, provides a sense of coherence and global direction. However, rigidly sticking to a plan results in a computer-based training (CBT) system that inflexibly follows a plan to the extent that it cannot accommodate unplanned questions, requests, or changes in goals. Such a system is also harder to adapt to new domains as so many decisions that are made in the plan are specific to the domain, tasks, presentation, curriculum, and expected trainee population.

In general, a balance must be struck between a totally reactive system, such as SOPHIE-I (Brown, Burton, & deKleer, 1982) and CBT systems with immutable plans that can only be followed. Dynamic instructional planning (Murray, 1990) is an approach to controlling intelligent tutoring systems that attempts to strike this balance by providing plans for global coherence, along with additional means for tracking and revising these plans when conditions require changes or allow improvements. The resulting control mechanism results in tutors that are not only more flexible in their interactions with trainees, but also across domains. They are easier to apply to new domains as the pedagogical, assessment, and control knowledge is, by necessity, abstracted into reusable modular units, rather than being procedurally compiled into a human-authored plan, as would be the case of a traditional computer-based tutor.

Dynamic planning allows the tutor to plan incrementally at multiple levels of tutorial abstraction and revise these plans to respond to new opportunities and to the trainee's interactions, including performance, questions, and requests. Dynamic control mechanisms such as MENO-TUTOR have provided similar multi-level plan selection mechanisms, but they lacked the ability to monitor global allocation of resources (e.g., time) and to patch or repair plans on the fly.

Our proposed tutorial system couples, for the first time, dynamic planning with true natural language processing capabilities. The result should provide a system where lesson planning and discourse planning are truly integrated for the first time. A particular advantage of this approach is that tutorial dialog is well suited to support 'soft skill' domains (e.g., leader training, decision-making). Most intelligent tutoring systems have avoided these areas, as it is easier to build subject matter expert models for precise, formal domains and test trainee knowledge about concrete items such as equipment components and their functions.

Dynamic Planning

The key to providing coherent and flexible instruction is a dynamic planning capability. This approach allows the tutor to have a plan but not be locked into it. Without a plan, the tutor may meander and not achieve instructional goals. Without flexibility, it may be ineffective in choosing the best approach for different kinds of trainees and reduce the motivation of trainees who are locked into a preset set of choices in controlling the tutorial interchange. The proposed Tutor has several levels of planning -- an overall global instructional strategy, topic sequencing and action planning, and discourse planning.

Global Instructional Strategy

The highest level of planning is the choice of tutorial strategies. For this effort, we have chosen to adopt a top-down approach -- introducing material to provide a 'big picture' overview, then teaching sub-skills, and then moving towards integration. Our top-down strategy is given in this example of teaching the troubleshooting of a complex hydraulic assembly (e.g., the lower hoist assembly of the Mark-45 naval gun turret described in Murray (1990).

1. First teach how the assembly operates normally. What are the parts and what is their function? The tutor questions the trainee's understanding before proceeding.
2. Next, teach a common troubleshooting approach. Check what cycle of operation the equipment is in based on the state of the solenoids. Then check that valves are operating correctly using pressure gauge tests. Again ensure the trainee understands this procedure via questioning.
3. Now practice on a variety of cases, helping trainees when they reach an impasse.

Topic sequencing (adapting to the individual)

The next lower level of planning is the choice and sequencing of topics for teaching, assessing, and practicing those topics. The topic sequencing will be determined by interpreting the networked node of knowledge units. This network can be traversed in different ways to produce different topic sequencing. We intend to support a fairly standard top-down approach where higher-level nodes are used to provide advance organizers (Reigeluth, 1987) and lower-level nodes correspond directly to procedural sub-skills or underlying conceptual units.

Selection of which node (instructional packet) to present next will be determined by evaluation of the trainee response. If a trainee provides an incomplete or incorrect response, then the Tutor will look for the next lower-level linked node indexed with the relevant information. If there is more than one relevant node, then the node will be selected based on other factors that quantitatively depict the relationships among the nodes and their links, and prior trainee performance.

Discourse planning: what to say and how to say it

Discourse planning is the lowest level of the overall integrated dynamic planning system that controls our tutor. Where tutorial instructional strategies can be viewed as top level planning, the discourse planning can be viewed as planning at a tactical level: what to say and how to say it. Evaluation of trainee input via the natural language understanding system will drive plan monitoring and student modeling. Questions, actions, and requests that indicate lack of trainee understanding result in revision in the discourse approach, topic activity, topic sequence, or tutorial strategy.

The tutor will answer trainee questions directly and fully based on the information contained in its knowledge base. After presenting material for a particular curriculum unit, the tutor will ask the trainee

questions from the problem set to ascertain trainee understanding. Based on evaluation of trainee responses, the tutor will specify intended speech acts ("triggers") that will be realized through the natural language generation system. Examples of the tutorial dialog triggers beyond question answering follow. More detail of how these dialog triggers will function is provided later in this section.

- *Pumping.* The tutor pumps the trainee for more information when an incomplete answer is given.
- *Prompting.* Tutor prompts trainee to fill in a missing word, phrase, or sentence.
- *Immediate feedback.* Tutor gives positive, negative, or neutral feedback after the trainee's response.
- *Hinting.* Tutor gives hints by presenting a fact or asking a leading question.

BB1 blackboard architecture: The dynamic planner

The overall control mechanism will be the BB1 blackboard architecture. Blackboards are well known as a sophisticated architecture for knowledge-based applications. The key features include independent knowledge sources, an agenda-based control mechanism, and a global blackboard that allows hierarchical representation of plans and evolving solutions. The BB1 blackboard is well suited to this application because multiple levels of planning can be easily represented, and the planning actions that occur at these multiple levels can be represented by independent knowledge sources. This modularity encourages reuse across multiple domains and facilitates incremental development.

BB1 provides a good fit to our tutorial system for a number of reasons. First, it supports meta-level reasoning through its use of a control blackboard. This control blackboard allows the blackboard to apply the blackboard model of problem solving to its own control problem. Practically, this can be used to monitor tutorial strategies and the overall planning process and to switch to more effective strategies or planning approaches when the situation arises.

Secondly, BB1 is built over a conceptual-graph knowledge representation based on Sowa's (Sowa, 1983) conceptual graphs. Such a knowledge representation is particularly suited to representing actions, events, and states and allowing knowledge sources that reason about actions, events, and states. More importantly, this knowledge representation is similar to the semantic network representation used by our dialog system. This similarity of representations should facilitate sharing of information, compared to knowledge representation mechanisms that adopt different approaches (e.g., frames and predicate calculus).

Finally, BB1 has a higher-level representation of actions, events, and states called *language frameworks* (Hayes-Roth, 1987). This is a pseudo-English like representation of a domain's operators. The blackboard system evaluates the utility of each possible action (planning, assessment, instructional, conversational, etc.) and chooses the best action at each time.

Integration of Tutor with Dialog System

Knowledge representation and dynamic planning

We will develop message-passing protocols to share information between the tutor's models (expert and student model) with the dialog system's knowledge representation and user belief model. Both the tutor's dynamic planner and the dialog system use conceptual graph representations which should facilitate communication. A full description of the knowledge representation schema is presented elsewhere in this proposal.

Discourse management

The multi-level hierarchical and opportunistic planning afforded by the dynamic planner will be used to track trainee interactions and drive the tutor's discourse triggers. An episodic memory of the tutor and trainee's utterances and actions will be retained so a common history can be discussed. Dialog between the trainee and tutor can take several forms. The trainee responds to a tutor's question or asks a question of the tutor. The tutor initiates a question or prompt, evaluates a trainee response, or answers a trainee's query. The tutor will be capable of a number of different speech acts, such as, pumping, prompting, splicing, hinting, summarizing, and explanation.

Question answering and explanation generation. If the trainee asks a question of the Tutor, then the Tutor's response will be formulated from the information in the Tutor's Domain Kb. The Tutor Domain Kb can represent conceptual or procedural information. Conceptual information items are definitions and simple relationships. Procedural information items are prescribed steps, preconditions for a particular path or goal, and exception rules. A description of how responses are generated from the knowledge base is described in the section, Dialog Agent System.

Evaluation of Trainee Input. If the trainee input is a response to a Tutor query, it is compared to an exemplar correct answer. Semantic and truth equivalence of the input compared to the exemplar is evaluated in two major steps. First the trainee response passes through the NLP module, performing a parse and initial semantic analysis on the sentence. Then it is evaluated against the exemplar answer for inclusion of all sub-facts and linguistic equivalence of each sub-fact.

A confidence rating of correct, incorrect, correct but incomplete is passed back to the Tutor along with identification of any missing items. The tutor planner will identify the nature of its reply (e.g., question, feedback, answer) and the key content items. This information will be passed to the language generator that will formulate the final Tutor utterance.

Tutor Initiated Dialog. Tutor initiated questions or prompts are triggered by the 'active event' in the tutorial strategy and key information is sent to the language generator. For example, if the active event is a move to a new lesson topic, the Tutor will send a trigger to the language generator indicating the discourse category ("introduce topic") and key content (e.g., node index: "command staff responsibilities"). The generator will fashion the appropriate introduction to the new topic, "Let's tackle a new topic, 'Command Staff Responsibilities'."

A tutor question is automatically presented following the material presented in an instructional unit. The trainee's response is evaluated and a confidence rating on accuracy and completeness is sent to the Tutor. Ratings will have thresholds for correct (accurate and complete), correct but incomplete, and incorrect (not accurate and not complete).

a) A correct evaluation will trigger the response generator to give affirmative feedback.

b) An incorrect evaluation will trigger the response generator to suggest the trainee review some underlying principles/steps.

- "What is the first step of the Military Decision-Making Process?"
- "First one establishes troops strength".
- "No. That's not right. I suggest we take a moment and review the seven steps in the MDMP."

c) A correct but incomplete response will trigger the response generator for a prompt for the missing information.

- "What are the intelligence disciplines?"
- "Human intelligence (HUMINT) and signal intelligence (SIGINT)."
- "Can you think of a third discipline?"
- "No"
- "What about radar and photographic sources?"
- "Oh, yes. Imagery intelligence (IMINT)!"

The Tutor will generate the prompt based on the information in the Tutor Domain Kb. For example, the Domain Kb will know that Intelligence disciplines break down into 3 sub-units of knowledge. The exemplar answer will have the 3 types and their names. The Tutor will expect an answer with the 3 sub-facts. The Kb will know that radar and photographic sources are examples of the missing sub-fact, Imagery Intelligence, and draw on those examples for the prompt. Imagery Intelligence and IMINT will be listed as synonyms in the lexicon and therefore understood to be semantically equivalent.

Authoring the Tutoring System

Tutorial Strategy. There will be some top-level parameters that can be adjusted (via slider bars) to modify the global characteristics of the tutorial strategy or the instructor can choose the default settings. These parameters will include assessment thresholds (i.e., how rigorously the trainee must perform before being able to move on to another topic) and level of tutorial assistance (i.e., degree of hinting and prompting).

Lessons (instructional packets). Associated with each of the knowledge units in the expert model is an option to author instructional packets. Instructional packets consist of lesson material and a Challenge (question-answer exercise). To attach instruction to a knowledge unit, one clicks on a button and the Instruction template pops up. The author types in the instructional narrative and clicks on a button to link it to a graphic, audio, or video file. Next the author types in the Challenge question and an exemplar correct response. The trainee's response will be compared to this exemplar response for semantic equivalence, accuracy, and completeness.

Knowledge Elicitation Tool for Tutor Domain Kb. Cognitive, procedural, and conceptual knowledge can be authored for the Tutor's domain Kb. We already have an operational knowledge elicitation tool for this purpose, called the Cognitive Analysis Tool © (CAT) (Williams, in press). It is a bit more difficult to describe the CAT tool in this text than it is to simply use it. A demonstration of the tool shows clearly the simplicity of the authoring and model building. The Office of Naval Research funded CAT and it was

used in a classified embedded training system for tactical decision making by Navy Tactical Action Officers. Army military officers and other subject matter experts (SME's) have already used CAT to build complex task models with minimal training (in most cases under an hour to learn the basics). It has also been used for distributed training environments, (e.g., bringing ships into harbor; tactics of a mechanized [tank] platoon).

CAT prompts the user to specify knowledge about a task or lesson objective as a set of goals and sub-goals. The result is an AND/OR graph of goals, conditions and actions. This set of rules is the knowledge base, also referred to as a cognitive task model. It forms a GOMS-like model, a lattice of interconnected knowledge nodes. CAT's output is a textual description of the nodes and rules of the model developed. We intend to use this output as input to the dynamic planner. The planner will be able to interpret the CAT model in a way that extends the original CAT model interpretation to include lower-level discourse planning and additional kinds of tutor re-planning.

CAT requires absolutely no programming skills. CAT automatically creates the node structure as the author fills in the fields of CAT's templates. The first step in authoring the Tutor Kb is to click on the Goal button and type in the main goal of the concept or procedure. The goal can be an objective, mission, aim, purpose, etc. Next, one fills in the template for the set of sub-goals or methods. If more than one method can be used to achieve a goal, the author can specify a set of conditions under which a particular method is used. Sub-goals are decomposed into steps. Steps can be specified as sequential, simultaneous, or in any order.

There are extensions that the author may choose to add to the model to get greater flexibility in the task model. Some of these options include *exception rules* and *impasses*. Exception rules suspend work on any goal and transfer control to a new goal (this departs from the typical GOMS tree structure). An impasse is a failed step. The author can identify steps that might fail, and select an impasse resolution option from a list of four alternatives. These are: 1) allowing the step to fail, 2) using another sub-goal, 3) defining a new alternative method, or 4) stating that the step can not fail.

CAT has a number of automatic aids to assist the author when building the model. For example, if the author types in one of the nodes, "Decide whether to replace the tire". CAT will prompt the author to fill in a template, "If ___, then do ___. Else do __". CAT also has some built-in conflict alerts. For example, if the author types in the exact same node more than once, then CAT brings this replication to the author's attention and asks whether it is intentional to duplicate the goal (step, or whatever).

CAT has the normal features of a Windows environment, with menus and menu options, selection of options with a mouse and point and click facility, and keyboard. There is an automatic help facility with an interactive context-sensitive help area at the bottom of the screen in addition to a user's manual and an on-line tutorial. When building CAT models, there are two modes of operation: a "guidance" mode and a "free-play edit" mode. The guidance mode is geared for novice and intermediate users because it dynamically guides the user through the process of developing the cognitive procedure and the appropriate constraints on each node. The free-play mode is geared for experienced users who want to build or edit models directly through the graphical interface. One simply clicks, drags, and drops the nodes onto the tree structure. Both authoring modes graphically display the tree structure and nodes with labels, or one can choose to view a fully textual representation.

TUTOR Screen Examples

Two examples of CAT authoring screens are provided in Figures 1 and 2. The top-most screen shows the authoring template and the bottom screen shows the networked structure that CAT automatically creates for the Tutor Domain Kb.

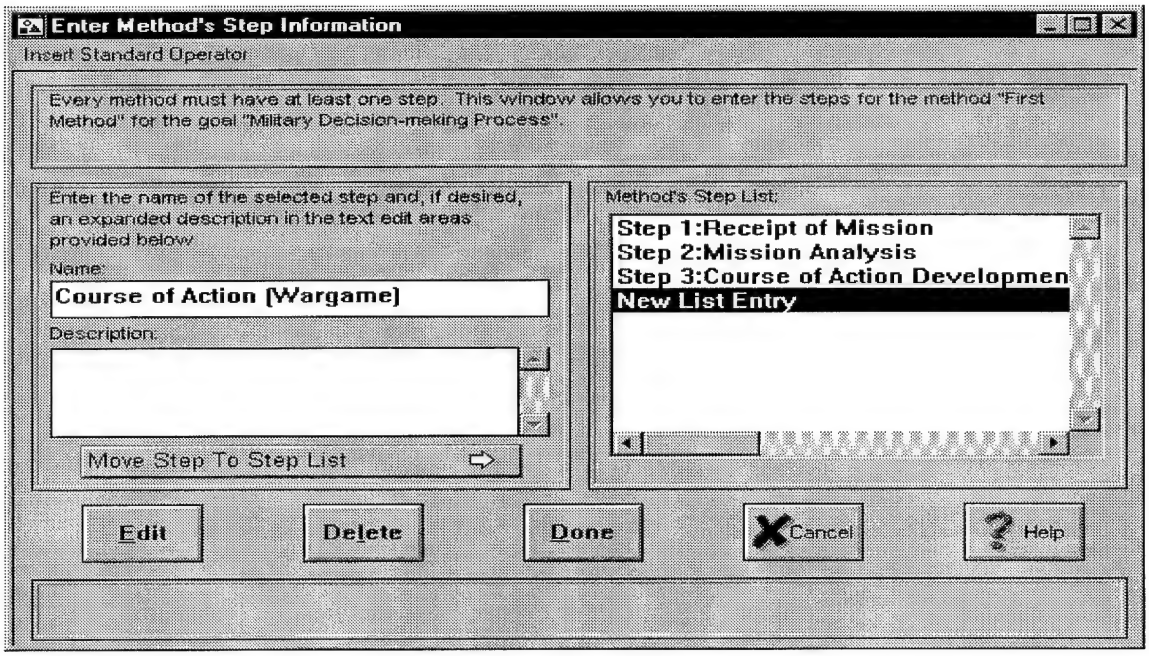
In Figure 1, the author is creating a model of the Military Decision-making Process. First the author selects a new model, types in the main domain topic. Then the author breaks down the topic into its top-level components or steps. Next the author would break down the major sub-goals (sub-topics) in each of these components, modeling the activities in Receipt of Mission, Mission Analysis, etc.

In Figure 2, the author is creating a model of a Brigade Commander's decision process to determine which of his staff officers to take to the Division OPORD briefing. The author has already typed in a field to always take the S2, S3, and FSO. Addition of other staff officers depends on the type of mission requiring a decision point. CAT has a template screen for IF-THEN-ELSE decisions that automatically comes up when the author types a step beginning with the word, "if".

Figure 3 shows the authoring screen for instructional units in the lesson. This example shows the instructional material for deciding which staff officers to take to the OPORD briefing. A Challenge question and example correct response is depicted.

Figure 4 shows how this instructional unit appears on the trainee screen. The trainee first reviews the instructional material, and then presses the "Done" button. Then Challenge question is presented. The trainee clicks on "Type" or "Speak", depending on preferred input mode. Spoken input appears in the Response box and can be corrected before submitted. The Tutor's response (e.g., evaluation, feedback) appears in the Tutor Response box. Again, the trainee can click on "Type" or "Speak", depending on whether they want to hear or read the Tutor's response. At any time, the trainee can click on "Talk to Tutor" button to engage in a 'side-bar' dialog.

Select New Model
Type Goal/Objective: Learn Military Decision-making Process
Type Top-level Steps: (see below)



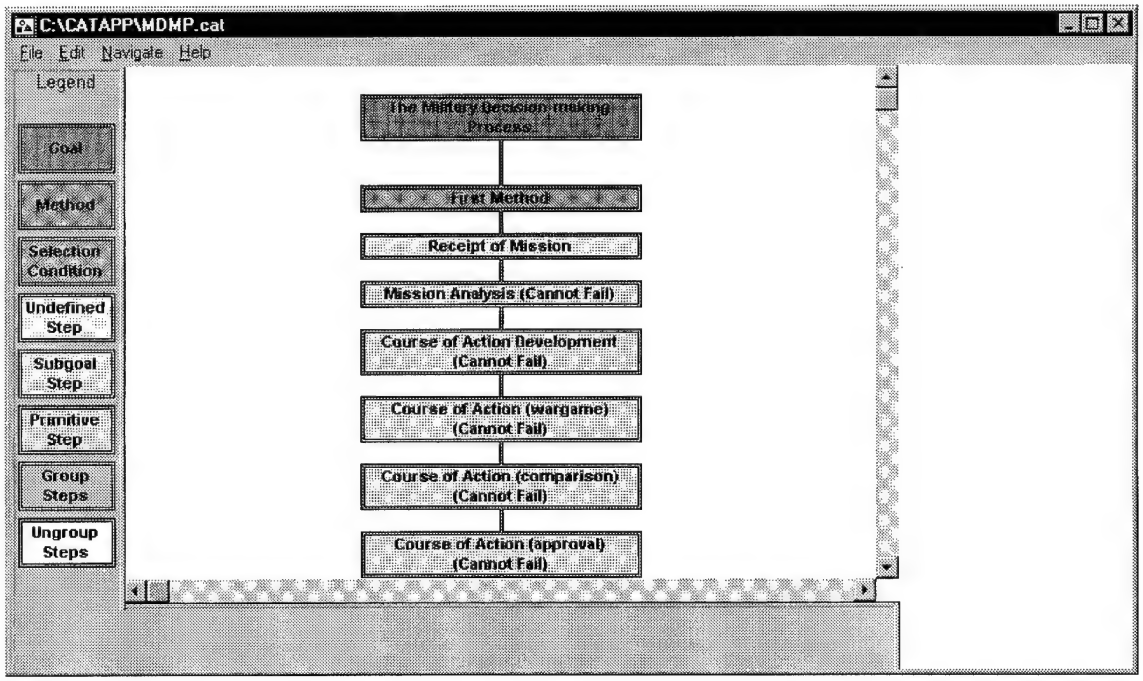
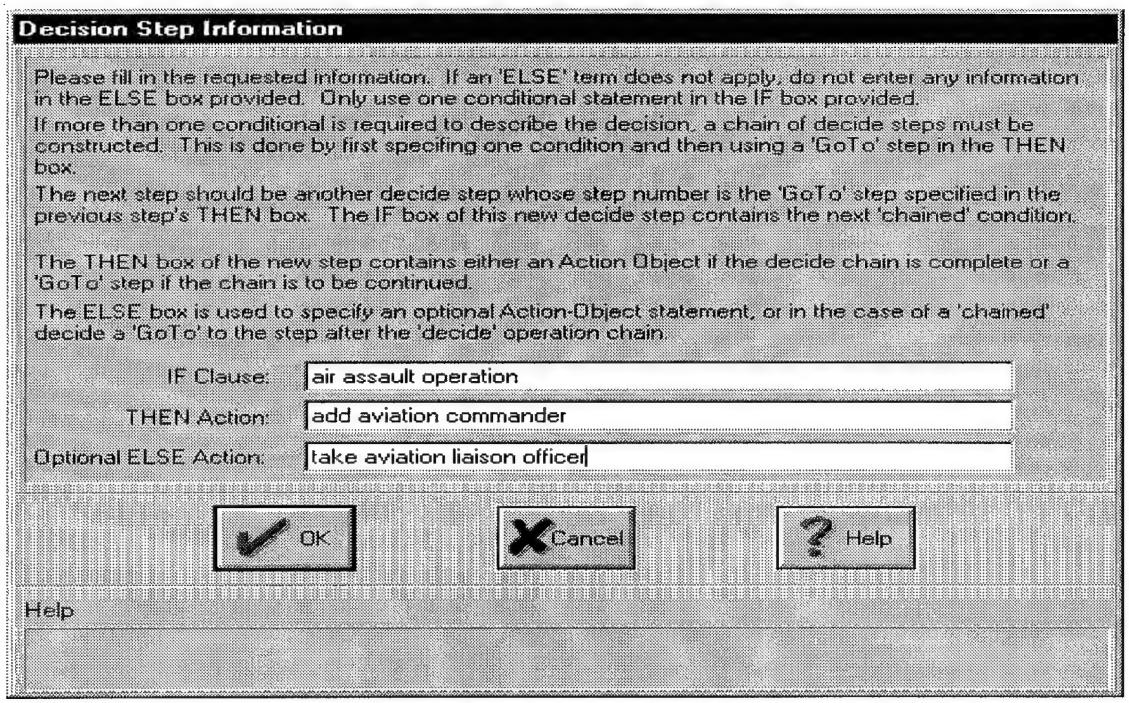


Figure 1. Authoring the Military Decision-making Process for the Tutor Kb. CAT authoring screen at top automatically builds the model shown below.



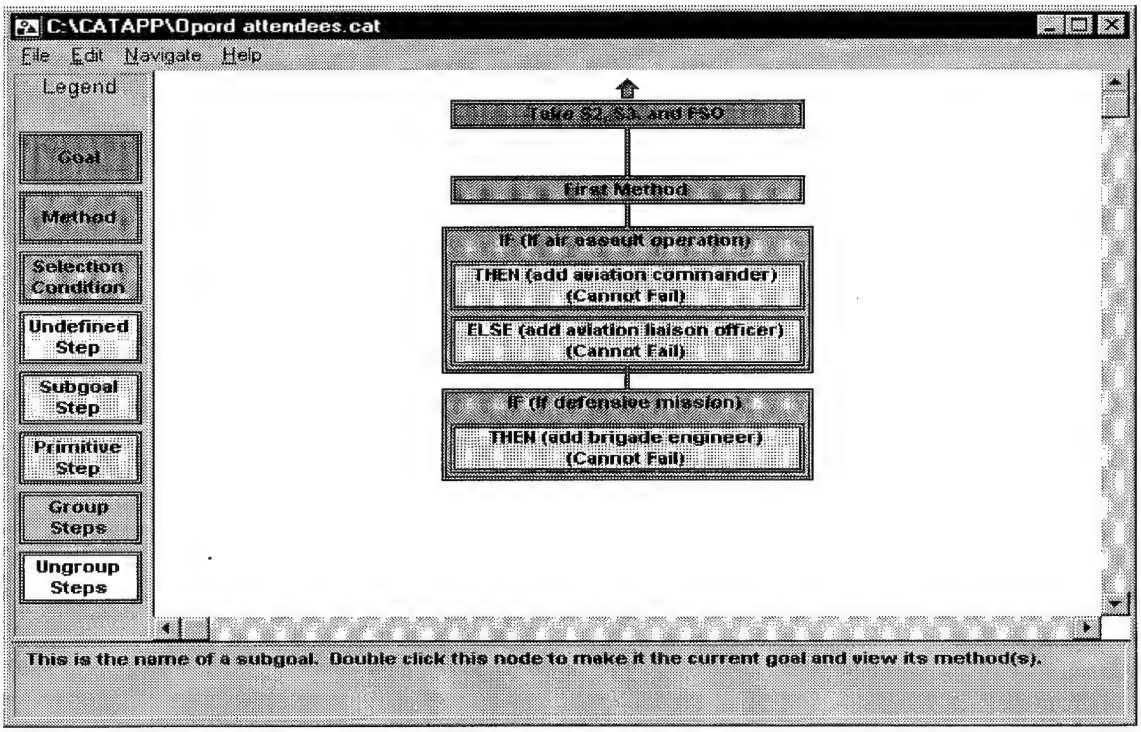


Figure 2. Authoring Brigade Commander process to determine which staff officers to take to Division OPORD briefing. CAT decision authoring screen at top builds the model shown below.

AUTHOR: INSTRUCTIONAL UNIT

Type Instructional Narrative in box below:

Division OPORD Briefing
Brigade Attendees

The brigade will normally receive a warning order (WARNO) that requires the commander to attend the division OPORD briefing at a specified time. The commander must determine who should go to the briefing with him. Tailored orders groups, clearly identified in the brigade SOP, will save notification time.

Normally he would take the S2, the S3, and FSO. If he has a defensive mission, he may add the brigade engineer. If the warning order reflects an air assault operation, he may want to add his aviation commander. If the aviation commander cannot attend, he may include his aviation liaison officer.

Add :

Graphic | Video | Audio

Type challenge below:

You are the Brigade Commander and just received a WARNO to attend the division OPORD briefing. It is a defensive mission. Which staff officers do you take with you?

Type an example of a correct answer below:

I'd take the S2, S3, SFO, and the brigade engineer.

Figure 3. Authoring an instructional unit in a lesson.

Instructional Material

Text and/or Graphic

Challenge

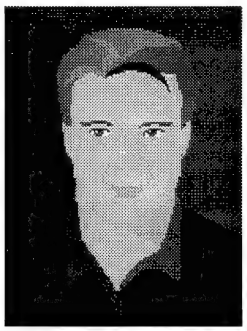
You are the Brigade Commander and just received a WARNO to attend the division OPORD briefing. It is a defensive mission. Which staff officers do you take with you?

Your Response [X] Type [ ] Speak

The S2, S3, and the FSO.

Tutor Response [X] Type [ ] Speak

Since it is a defensive mission, isn't there another Officer you might take along?



[ ] Talk to Tutor

Figure 4. Sample Trainee Screen. Instructional material is presented first, then the Challenge. The trainee can ask or answer questions at any time.

The Dialog Agent System

Overview

Giving voice to the frustration of a generation of ITS researchers, Mark Miller once declared that "natural language interfaces are a diversion" in the field of Intelligent Tutoring Systems [Psotka et al. 1988:407]. The difficulty with this dictum is that (written and spoken) natural language has been the preferred interface between teachers and learners throughout all of human history. Could ITSs be so very different from everything that has gone before? Or is the real problem not that natural language is irrelevant to the automation of the instructional function, but just that it is very hard to do?

Our Dialog Tutor project transcends this conventional wisdom, distinguishing itself from other ITS in its reliance on Socratic, peer-to-peer, and other modes of unconstrained conversation as a primary instructional delivery vehicle. In our environment, trainees converse as they would with English-speaking human colocutors — engaging in questions-and-answer sessions with virtual tutors, or talking with simulated co-workers, interviewees, or even trainees in role-playing scenarios. In all such cases, dialogue-enabled agents respond appropriately to the trainee's input, holding up their end of the simulated dialogue with extemporaneous responses on any defined instructional domain.

In this section, we explore the technologies that comprise the Dialogue Agent System (DAS) — parsing, knowledge representation, discourse management, language generation, speech processing — as well as the DAS authoring facilities which will enable non-programmers to create and customize such agents to meet new instructional needs. First, we describe where each of these technologies fits into the overall functional architecture of the Dialogue Agent System.

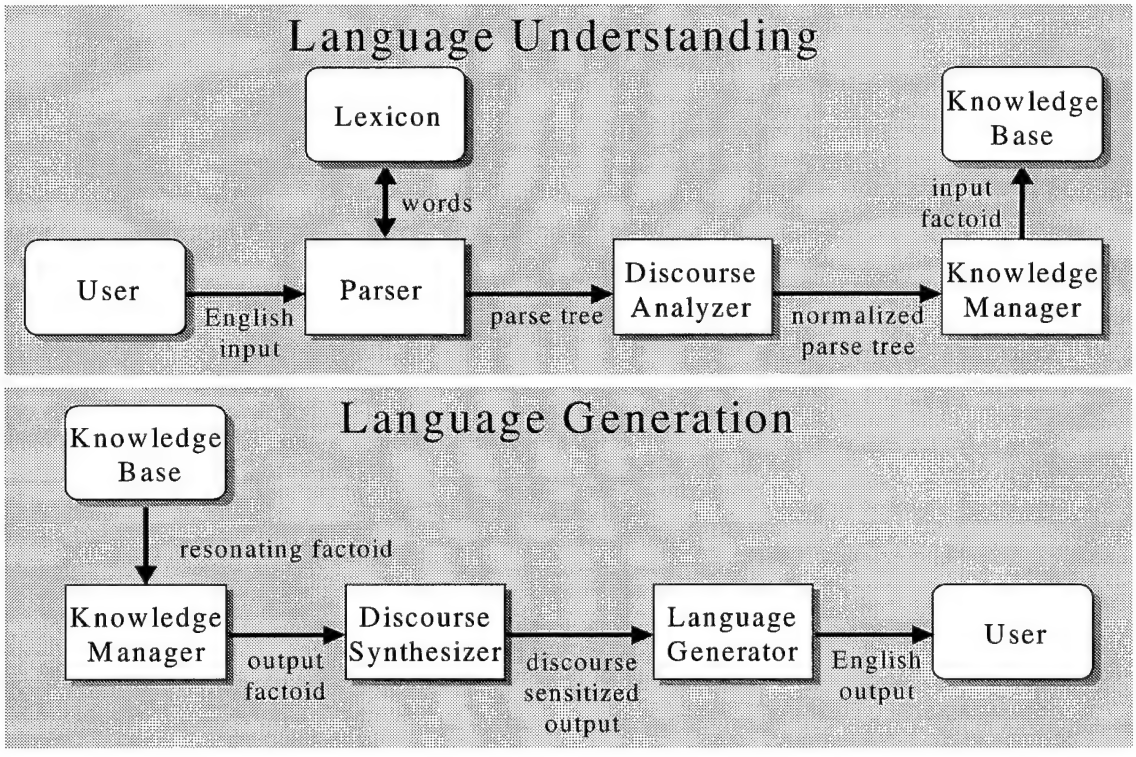


Figure 5. A process diagram of the Dialog Agent System.

Input Parsing and Lexicon. The trainee's textual input is passed to a parser, which analyzes the input sentence into a parse tree, a set of relations which formally specifies who did what to whom and other pertinent syntactic and low-level semantic information. In order to complete this low-level analysis, the parser relies upon several sub-components including a lexicon containing the key analytic features of words in the lesson domain, and a morphological analyzer which, among other tasks, decomposes words like "morphological" into "morphology" + "ical."

Semantic Interpretation and Discourse Analysis. Next, the output of the parser is mapped into a semantic case-frame. As part of the mapping process, the results of syntactic analysis are enhanced with contextual information from DAS's discourse-management technology, enabling the resolution of anaphora, ellipses, etc. In addition, the semantic transform itself resolves synonyms, polysemes, and paraphrases. At this point, what began as free-form learner input has been reduced to one or more propositions (or "factoids") expressed in a language-independent formalism.

Knowledge Base Management and Factoid Validation. The factoid is now passed to DAS's knowledge manager, the subsystem administering the knowledge bases (KBs) which represent both the current state of the world and the "mindsets" — simulated memories, beliefs, and motivations — of the various agents populating it. It is by virtue of these persistent models of the world and of an agent's own psyche that the agent's eventual response, though improvised, can nonetheless exhibit the consistency and coherence that we humans expect from our conversational partners. To begin the process of generating that response, then, the knowledge manager's inferencing engine attempts to match the incoming factoid against the contents of the currently active mindset. To the extent that some KB entry can be found to "resonate" — immediately or inferentially — with the input factoid, the learner's original input is deemed to be wholly or partially valid.

User Model Maintenance and Performance Monitoring. In addition to managing the knowledge bases representing its model of the world and of the currently active agent's mindset, DAS also maintains a knowledge base dedicated to modeling the learner's beliefs. As soon as it has been validated against the "objective" world-model, the input factoid is inserted in this user-belief model as well. As this adaptive learner model evolves over the course of the conversation, "stealth watch" processes monitor its state in the background, looking for the sort of deviations from the norm that might mandate a tutorial intervention or shift in pedagogical strategy. More immediately, and depending on conversational context, the insert may also serve to trigger a corresponding discourse gambit. For example, if the transaction fails because the input factoid already exists in the learner model, DAS may have the currently active agent remind the trainee that, say, this question has already been asked and answered.

Meta-Dialogue and Turn Taking. The last operation DAS performs prior to formulating a candidate output response is to check whether anything in the input (at the syntactic, semantic, or discourse level), or in the simulated environment, would cause the conversational initiative to pass from the agent to the user, or vice versa. This is key to response formulation because the form and content of the agent's next dialogue move depend in part upon whether DAS is actively pursuing its own conversational agenda, or reacting passively to the learner's. As suggested immediately above, turn-passing can be triggered by input that skews DAS's user-belief model so far out of tolerance as to drive TUTOR to assume control of the dialogue (if it does not already have it). But there are other, less catastrophic scenarios leading to the same outcome — e.g., answering a question with a question can signal the learner's desire to take the lead for a few turns.

Response Formulation. Once it has assessed the validity of the input, and established its own conversational stance, DAS is finally in a position to formulate a candidate response. One contributing factor is, of course, the degree of validity of the stimulus factoid. At the simplest level, this determines whether the agent is inclined to agree or disagree with the learner's input. More interestingly, an input's partial "resonance" with a given KB entry can cause the agent to volunteer new information in reply.

Discourse Synthesis and Language Generation. The candidate-response factoid is then run back through a discourse-management filter to screen out any of its elements which are not germane to the current discussion, and to accord appropriate emphasis to those which are. The resulting discourse-sensitive factoid is then fed into DAS's language generator, which performs a mapping from semantic to syntactic case frames, applies surface transforms (e.g., pronomialization) consistent with the focus of discourse, and formulates an output string for delivery to the learner.

This bundle of generic DAS technologies can, in turn, be used to fashion a variety of dialogue-based agents, each with its specific role in the Tutor system:

Virtual Tutors/Mentors to serve as a conversational "front end" to the Intelligent Tutoring capabilities;

Conversational Agents to enact the parts of virtual coworkers, superiors, subordinates — even virtual trainees for the learner to instruct — in dialogue-based role-playing scenarios;

Stealth Watchers that monitor the user-belief model in the background, intervening with remediation if updates cause the model to stray out of tolerance range;

Author's Assistants which support the creation of new instructional material by engaging the author in dialogues regarding the categorization of new words and concepts, the clarification of newly input factoids, etc. With this functional overview as context, we now turn to a more detailed, technical discussion of the enabling technologies.

Details of DAS Technical Approach

Parsing. In order to determine *who did what to whom*, DAS relies on its symbolic parser. This parser is capable of handling a broad range of clausal English constructions. For example:

a. SSG Bozo gave an insult to PFC Aiken.
b. He insulted PFC Aiken.
c. PFC Aiken was insulted by SSG Bozo.
d. It was PFC Aiken who was insulted by SSG Bozo.
1. What did SSG Bozo harass PFC Aiken with?

In *(a)* the semantic relations of the sentence are canonical. SSG Bozo is the *who* (the semantic "actor" or "agent"), an *insult* is the *what* (the semantic "patient" or "theme"), and PFC Aiken is the *whom* (the abstract semantic "goal", a polymorphic semantic role often realized as either a "beneficiary" or an "experiencer"). In *(b)* there is no overt *what* (the *what* has been incorporated into the verb as in "SSgt Bozo insulted an insult to Pfc Aiken"). In *(c)* the *whom* precedes the *who* (a "passive" construction). In *(d)* a passive construction is embedded within a "cleft" sentence, *It was X*. Although in *(b)* it is the task of the dialogue manager to identify who *He* is, in *(d)* it is the task of the parser to resolve *who* to *Pfc Aiken*. *(e)*, a question, illustrates that the order of the principal semantic elements can be permuted, here to *what-who-whom*. There are many more permutations of these fundamental elements and the many "oblique" elements of sentences (e.g., *when, why, how, where, if, because*). The parser must disentangle all of these before further processing is possible.

The parser recursively builds upon these clausal capabilities to perform metaclausal analysis, handling such constructs as subordinate/relative and coordinate clauses. Symbolic parsers can be divided into bottom-up and top-down types. Context-free bottom-up parsers are more efficient than context-sensitive top-down parsers if, and only if; 1) all interpretations of a sentence are of interest (e.g., *Flying airplanes can be dangerous*), and 2) there are no homonyms or polysemes in the sentence.

It is normally desirable, however, for a parser to be context-sensitive, so as to return only the most relevant interpretation (e.g., only *The flying of (or on) airplanes can be dangerous,* not *Airplanes which are flying can be dangerous*). Relatively uninflected languages like English are also rich in homonyms (e.g., *return* (v) vs. *return* (n)) and polysemes (e.g., *rich* (fatty, caloric) vs. *rich* (affluent) vs. *rich* (abundant)). Because conditions (1) and (2) are rarely encountered in English natural language processing tasks, top-down symbolic parsers are much more efficient in practice.

Top-down symbolic parsers can, in turn, be divided into two broad types: definite clause grammar (DCG) parsers and augmented transition network (ATN) parsers. Simple definite clause grammars consist of simple rules and are easy to author for small domains. Simple ATN grammars are more code-like and are harder to write, but as domains become large and grammars approach wide coverage, DCG grammars become harder to maintain than ATNs.

In light of the preceding considerations, DAS has adopted a Generalized Transition Network parser [Loritz 1993], which is an ATN parser extended by several DCG-like enhancements including shift-reduce mechanisms [Sato 1988], "fitted" bottom-up parsing [Jensen et al., 1993], deterministic "look-ahead" [Marcus 1980], a finite-state morphological transducer [Koskenniemi 1983] which is "cascaded" [Woods, 1980], and a gap-threaded hold list [cf. Alshawi 1992].

The DAS parser currently parses with better than 95% syntactic accuracy on a wide range of constructions including English statements, questions, and imperatives, including passives clauses, inchoative and inceptive clauses, essive clauses, existential clauses, relative clauses, participial clauses and phrases, subordinate clauses, cleft sentences, pseudo-cleft sentences, and various gapped constructions. The grammar therefore meets the definition of a wide-coverage grammar. Parser error, when it does occur, is usually attributable to deficiencies in the lexicon.

Lexicon. An adequate lexicon is a major problem for all parsing systems. Large, preconstructed online dictionaries can be adapted for parser use, but in this case there are at least three reasons why bigger, by itself, is not better. First, the lexicon ultimately needs to be coordinated with the ontology, and the larger the lexicon, the more difficult this task becomes. Second, as noted above, homonymy and polysemy seriously degrade a parser's performance. For this reason it is undesirable to have a large lexicon which is deeply coded with many out-of-domain word senses and subsenses. Third, "training" almost always entails the learning of new technical terms and specialized jargon. As a result, no pre-constructed lexicon, no matter how large, is likely to have the most important words for the (instructional) task at hand.

Instead, the DAS parser operates with a core lexicon of the 4,000 most frequent English words, modeled on an adaptive resonance neural network (Grossberg 1986, Loritz *in press*), and a main supplementary lexicon of the next-most frequent 16,000 English words [Carroll et al. 1971]. These combine to cover, in appreciable depth, some 30,000 word senses. Additional, domain-specific supplemental lexicons can also be called if an input word is not found in these master lexicons.

The upshot of all these design considerations is that the DAS lexicon must be authorable. The DAS lexicon was designed for authorability from the ground up. To simplify lexicography, its lexicons were organized as semantic networks: taxonomies into which new terms can be added and from which default lexical features can be automatically inherited [Loritz 1993]. We will return to the issue of lexicon authoring as part of our general discussion of authoring.

Discourse Management

Having retrieved the definitions of individual words, and parsed the syntax of the sentences containing them, DAS focuses on the next higher unit of analysis: the discourse itself — the multiple sentences comprising multiple dialogue "turns" which together establish the broader context in which interpretation must take place. At this level, the equivalent to the parse of a sentence is the "focus of discourse" for the conversation as a whole.

The principal phenomena to be addressed here are:

- **agreement-signaling** — signaling corroboration or disagreement (e.g. *That's right* vs. *Hmm, well ..*) and **coordination** — elaborations to what the previous speaker has just finished saying, often as an incomplete sentence (e.g., *...and the repair costs too!*);
- **topic and topic-shift signaling** — the cues used to identify the current topic, or to alert the listener that the speaker is about to change topics (e.g., *Going back to your previous point, ...*);
- **anaphora** — constructions that need to be resolved in terms of antecedent references, principally: personal pronouns (*he, she, it*), propositional anaphora (*I know that*), one-anaphora (*One of our aircraft is missing*), deictics (*this, that, here, there, now, then, you, me*), and definite descriptors (*What did the doctor say then?* referring to a previously mentioned physician);
- **ellipsis** — sentence fragments often occurring as requests for an elaboration of the preceding statement (*How?, Why?, With whom?*).

Without the ability to cross reference *he* or *she* to some concrete, named individual, the interpretation of a sentence's meaning would grind to a halt for lack of specifics. Ongoing research and development has already yielded a toolkit of discourse-management heuristics for the resolution of anaphoric, deictic, and other meta-sentential reference.

Knowledge Representation

The work of lexical, syntactic, and discourse analysis culminates in an unambiguous rendering of linguistic input into its component objects and relations.

The Representational Hierarchy. The DAS knowledge-representation schema is a structured hierarchy, but is not a simple inheritance hierarchy. Our schema can represent a wide range of factoids, from simple propositional relationships and declarative clauses to more complex relationships and concepts. For example, it can accommodate the simple declarative statement, "Joe is in the motor pool". It can also represent binary relationships such as, "Joe was not at the meeting because he was in the motor pool". The knowledge representation structure can accommodate a wide variety of complex relationships such as, deterministic and probabilistic logical and relational operators, implication, universal and existential quantification. For example, the schema has the ability to capture common-sense propositions like "no

one can be in two places at once."[1] A more extensive discussion of simple factoids (under the name "postulations") may be found in [DeSmedt 1995].

Representation and Dialogue. The evolution of DAS's dialogue capabilities parallels this representational hierarchy: the factoids representing the simple relationships are sufficient to handle most scenarios in which the user asks questions and the virtual agent simply responds. Such user-initiative interrogations are, in fact, the central plot device in the *Herr Kommissar®* German conversation tutor, an early application of the DAS technologies to the instructional domain. Factoids representing binary relationships support closed-context turn-taking, such as might be driven by a query focused domain-specific tutorial dialogue capability. Open-ended mixed-initiative dialogues, on the other hand, requires all the expressive power of the more complex representational schema.

This is because, in order to convincingly take the initiative in a conversation, the computer must be endowed with something resembling adaptive conversational goals — as well as with a means of formulating plans to realize those goals via conversational stratagems. Since these "motivational" relations, in turn, can be modeled as a set of backward-chaining implications, they can be expressed in our representation schema. The end-result is that the computer, no less than the user, will have a set of issues that it "wants" to resolve by means of the dialogue, and the wherewithal to take the initiative in resolving them. Provided one of the computer's "sub-motivations" is an adaptively-controlled turn-taking constraint (a.k.a. "politeness"), genuine mixed-initiative dialogue will be the result.

Work already underway will result in the development of a first-generation representation of simple propositions. These representations will provide only a subset of the above features — sufficient to endow an agent with a single motivation at any point in time, but not to mediate among multiple competing motivations. A DAS agent whose conversational motivations are based on this level of representation will have, literally, "a one-track mind."

Performance Evaluation/Intervention

One class of Dialogue Agents seldom participates in the dialogue itself. These are the "stealth watchers" responsible instead for monitoring what the progress of that dialogue reveals about the learner's comprehension of the subject matter. After the Tutor presents some instructional material, it then asks the trainee a Challenge question to ascertain trainee mastery of the material. The trainee's answer is compared against the DAS knowledge base, sentence-by-sentence, for logical consistency. Where sentences and clauses are ordered by temporal or causal connectives (*after, because,* etc.), these orderings are likewise tested for consistency against matching representations in the DAS knowledge base. A global consistency metric can be returned to the ITS and/or specific mismatches reported back to the learner (e.g., "You do not seem to know *when* your summary report is due to the commanding officer.")

Specific, discrete-point questions may be asked. For example, the Challenge might be: "What are the intelligence disciplines?" The trainee is expected to reply with specific information, which the instructor authors in the form of exemplary answers. "The intelligence disciplines are human intelligence, imagery

---

[1] More precisely: "for all x, y, z, where (x is a person) and (y is a location) and (z is a location), if (x is at y) and (y is not equal to z) and (y does not contain z) and (z does not contain y), then (x is not at z)" That is, this deceptively simple commonsense rule requires eight atomic propositions, five relational operators (six, counting the "not"), an implication, and three universal quantifications.

intelligence, and signal intelligence." DAS evaluates strict goodness-of-fit between learner answers and exemplary answers. In so doing, DAS must accept variants like the following as correct:

- "The intelligence disciplines are signal intelligence, human intelligence, and imagery intelligence." [permuted terms]
- "The intelligence disciplines are SIGINT, HUMINT, AND IMINT." [hyponymy, hypernymy, synonymy relations]

While, at the same time, rejecting variants such as these:

- "The intelligence disciplines are signal intelligence and human intelligence." [missing term]
- "The intelligence disciplines are SIGINT, HUMINT, AND BREATHMINT." [wrong term]

DAS's general technique for evaluating strict equivalence to exemplary answers proceeds very much as does the analysis cycle in its processing of dialogue generally:

- Reduce the Challenge question, learner answer, and exemplary answer to their logical form (parse tree) equivalents;
- Resolve anaphora, sentence fragments, etc. contained in the learner answer by reference to the parse tree for the Challenge question, to yield a "normalized" parse tree for the learner answer;
- Map all lexical entries in the parses for the learner and exemplary answers to concepts in the DAS ontology, resolving hyponyms, hypernyms, and synonyms to identical concepts in the process;
- Map the resulting "conceptualized" parse trees for the learner and exemplary answer to their knowledge-structure (factoid) equivalents;
- Attempt logical unification ("resonance" assessment) of the resulting factoids for learner answer and exemplary answer.

The typical use of such discrete-point question exercises is to return a 1/0 score to the ITS, but DAS can also transform a partially-resonant answer so as to provide specific feedback to the learner (e.g., "BREATHMINT is *not* one of the intelligence disciplines.")

Authoring the Dialog Agent System

Authoring the Dialog Agent System involves two tasks: (1) the creation of standalone agents (especially, conversational agents) and their mindsets, and (2) the integration of information from the Tutor Kb and Instructional Units into DAS's knowledge bases. To help with authoring the knowledge representation, we provide an Authoring Assistant (AA) agent that engages in question-answer dialog with the user.

Authoring Conversational Agents. All of DAS's primary agent types -- tutors, conversational agents, and stealth watchers -- can be authored by non-programmers, using plain English as well as familiar GUI interactions. But the conversational agents featured in role-playing dialogues in skill practice scenarios are more illustrative of the process, since they typically require more in the way of character-personality authoring than do tutorial or stealth-watch agents. In role-playing exercises, conversational agents may assume various *personae: male or female, commissioned or noncom, cooperative or uncooperative, etc.* These personality traits can be authored with radio buttons and slider bars. Each unique persona can then be given a mindset -- a collection of factoids representing the agent's memories, beliefs, and specialized knowledge.

Character Configuration. The first step in creating a new dialogue agent is to establish some basics of characterization. All of the agents are named entities. Authoring a named entity involves creating a new character, and assigning it both an entity type and a name. The entity type is selected from a cascading menu-list of the concepts in the ontology's entity sub-tree (roughly corresponding to the class of common nouns). Names are assigned by choosing items from list-boxes, with the lists' content and structure being sensitive to entity type — e.g., an entity typed as *female_human* is named from three list-boxes for given, middle, and surname, the contents of the first two being constrained to women's names, while an entity classed as *male_golden_retriever* is named from a single list-box of male canine names ("Fido," "Rover," etc.). If the author fails to finds the desired name and/or entity-type available, new ones may be created via the same processes of lexicon and ontology authoring as described below.

Once established, the agent's character may be endowed with attributes. Of the attributes which may be authored, the most important for DAS's purposes are the "personality" traits which affect a character's conversational style(s). GUI sliders will enable the author to set parameters along sliding scales for, say, courtesy (from *rude* to *obsequious*), disposition (*despondent* to *ecstatic*), attitude (*obstructionist* to *helpful*), or loquaciousness (*reticent* to *voluble*). These settings will be taken into account at the point of DAS response generation, and reflected in the final form of the utterance. As basic idiosyncrasies, these attributes may be weakened or strengthened, but never entirely eliminated, by the force of "external" circumstances as represented by the current state of the world model.

Finally, the character configuration will also enable the author to assign vital statistics — age, nationality, hair color, etc. — to a character using GUI controls. Such characteristics, however, are not in fact immutable traits but fluents, and are consequently transformed into their factoid equivalents prior to storage. For example, the character's age attribute is stored internally as a world-state factoid of the form "*X* was born in the year *Y*".

MindSet Authoring. Authoring an agent's mindset involves editing an individualized DAS knowledge base specific to that agent. The contents of each mindset KB represent all of an agent's memories, beliefs, specialized (as opposed to common) knowledge, and motivations, stored in the knowledge representation. Wherever appropriate, as in the case of creating or editing the knowledge representations, DAS's dialogue capabilities are invoked to render the process of mindset authoring conversational in nature. Far from requiring anything like programming skills, entering such simple factoids into a mindset resembles nothing so much as a drama coach's rehearsal with an improvisational actor.

1. The author keys in one or more plain English sentences incorporating the memory or belief to be imparted to the agent. Since the full power of DAS will be brought to bear on interpreting this input, it may of course contain anaphora, ellipsis, paraphrase — even grammar and spelling errors.

2. Assuming no unfamiliar words or concepts are encountered (a circumstance addressed below), DAS transforms the author's sentences into factoids, just as it would for a trainee's input when operating in normal mode.

3. Since simple and binary factoids can accommodate more information than is usually provided in any single English sentence, the Authoring Assistant (AA) opens a dialogue with the author intended to fill in the missing data. For example, if the author enters the sentence "SSG Bozo insulted PVT Craven yesterday," DAS will direct the AA agent to inquire "At what time?," "Where?"and the all-important "Why?"

4. Once a well-formed factoid has been fashioned, DAS subjects it to a consistency check against both the current contents of the agent's mindset and the world knowledge base, using DAS's standard resonance-testing procedures. If the candidate factoid turns out to contradict either common sense or one of the agent's existing beliefs, the AA will point this out to the author.

5. DAS will not, however, block the insertion of a contradictory factoid into an agent's mindset if the author confirms the transaction. Rather, the ability to enter contrafactual information is the means by which to author the mindset of a lying, misguided, or insane character.

6. Finally, once new memories, beliefs, or specialist knowledge has been authored for a character, the system should allow the author to enter a test-mode, in which he or she can converse with that character to determine whether its responses are now in keeping with the author's objective. The ultimate goal is a system where one "rehearses" a character and then "auditions" it for the role it is to play. An apt model for such authoring would be the sort of sessions in which intelligence operatives are first briefed on, and then grilled on, their "legends."

Editing an existing factoid is accomplished by a nearly identical process, the only difference being that the factoid in question must first be selected from a list of the mindset's current contents before the authoring conversation can begin.

Taxonomy Authoring. Adding a new concept — and especially a new class-subtree to the ontology — involves more than merely inserting another proposition in the knowledge base. The author will be shielded from this complexity via a simple question-and-answer dialogue with the AA.

First, the AA establishes where the new subclass or individual concept attaches to the existing taxonomy by asking the author for the name of the class to which the new concept belongs. If the author names a class which is also not in the ontology, the AA presents the choice of either recursively naming a class to which *that* class belongs, or choosing a base class from a cascading (hierarchical) menu.

Second, once the new subclass has been linked to a base class, the AA asks the author to specify in plain-English how the new class differs from its parent. The author's response is transformed into one or more factoids which are then associated with the newly installed concept, for use in subsequent differentiations.

Lexicon Authoring. After a new concept has been added to the ontology, the corresponding English word must be added to the lexicon. Of course the most common 30,000 English word senses will already be in the DAS lexicon, but technical terms (e.g., SIGINT, *antistatic* pad, *browser*) and specialized senses of common words (e.g., *drive=disk drive)* are not reliably found or defined even in print or online dictionaries. The DAS authoring interface will make lexical authoring easy. It will ask the author to use the new term in a sentence (or take a sentence just-authored) and propose a set of likely lexical features.

browser
can be used in the following phrases (check only those that are grammatical):
☑ many browsers
☐ much browser
☐ a browser is a person

Integrating CAT output with the Dialog System

Instructors first author the Tutor KB and Instructional units via CAT (described in the ITS section). This creates the domain knowledge structure and instructional material -- this is where most ITS leave off. However, for the Tutor to dialog about the material, it must be integrated in the DAS (lexicon, dialog ontology, etc.). The author clicks on a "Submit to Dialog System" button, and CAT automatically passes the textual information to the DAS. DAS then performs lexical retrieval, syntactic analysis, and semantic interpretation on the text to create the corresponding factoids. If DAS encounters no unknown terms or missing information during this process, its knowledge manager then automatically adds the new factoids to the knowledge base being created for the lesson, storing with each one a unique index pointer provided by CAT. As each factoid is entered into its KB, the knowledge manager assigns it a unique DAS index which is subsequently passed back to CAT. At the completion of this interchange, every factoid in the DAS lesson-KB will bear a pointer to the corresponding CAT node, and vice versa. It is by reference to these indices that each subsystem keeps the other in synch. If unknown terms or concepts are encountered, then the author is prompted and goes through the DAS authoring procedure described earlier.

Spoken Interface and Talking Head

The SBIR requires that the system's simulated dialog partner be able to recognize spoken input and produce spoken output accompanied by an animated talking head running in a Windows 95/NT environment. Microsoft has recently released version 3.0 of its Speech Application Programming Interface (MS-SAPI). This defines standard speech recognition and text-to-speech interfaces for Windows applications. It also defines a "TTSMOUTH" interface, a common specification for control of "talking heads". Therefore, MS-SAPI compliance is an important consideration in our selection of the components.

In our Phase I proposal, we had proposed to include the speech recognition technology (*CSLUC*) from the Oregon Graduate Institute (OGI) and *Baldi*, a "talking head" developed at UC Santa Cruz, which is lip-synched to the *Festival* text-to-speech (TTS) system from the University of Edinburgh. Recently these three academic units have joined in a joint commercial venture called Fluent Technologies. While Fluent is doing some excellent research and development, all the components of their package do not currently meet our interface and performance specifications on a Windows 95/NT platform.

Therefore, we decided to conduct further evaluations on several different systems during our Phase I period. We evaluated the feasibility of using speech recognition and synthesis technologies from IBM, Dragon, Lernout & Hauspie, Fluent, and Entropic, and their subsidiaries. Evaluations were based on performance, interface and platform requirements, ease of use, technical support, and cost. We conducted our tests on a P200 machine running Windows95 with 64MB of RAM. A summary of our evaluation is presented, followed by a product comparison table at the end of this section.

We concluded that IBM's *Via Voice* is the best overall package choice at this time, both in terms of performance and cost. *Via Voice* is designed to be speaker independent and showed about a 96% speech recognition accuracy in our tests. The *Via Voice* package had the strongest TTS engine in the group that we evaluated. It supplies eight pre-configured voices, as well as an easy authoring interface for creating a wide range of variable additional voices. The IBM developer kit also includes Virtual Voices, its interface between the Microsoft interface and eight talking-head "actors" (man, woman, child, and 5 caricature actors) with six preset eye expressions that one can be controlled independently of the bottom-frame lip expressions. The *Via Voice* package was easy to install and use, met our interface

requirements, and in addition, IBM provided excellent technical support. *Via Voice* costs $100 per workstation.

The spoken interface is simply just that - an interface to our dialog system -- therefore, we plan to initially develop for the broad Microsoft SAPI standard, and to only make our final vendor selection near the end of the first development year in 1999. Since this is a rapidly evolving technology, waiting will allow us to select the best product on the market at that time.

Continuous Speech Recognition vs. Continuous Dictation

The Holy Grail of speech recognition for years was referred to as "Continuous Speech Recognition", the ability to recognize all words spoken from a large vocabulary (not just "word-spotting" a few key words) even though they are normally connected and run-together in natural speech (Schafer and Rabiner 1975). As a contrastive example, Lernout & Hauspie's Kurzweil large-vocabulary dictation products still only support discrete, "isolated word" recognition—where each word is separated by a small pause (although one has to read down to the fine print to confirm this fact).

Lately, many systems capable only of recognizing continuous speech within relatively limited vocabularies (several hundred to several thousand words) have taken to implying that they are "continuous speech recognition" systems. For example, the Fluent and Entropic systems both include low-level Hidden Markov Model (HMM, *cf.* Rabiner and Juang 1993) toolkits, the basic technology needed for true continuous speech recognition, and so are theoretically capable of delivering large-vocabulary continuous speech recognition. However even with a HMM toolkit, the task of creating a large-vocabulary model for continuous speech recognition is daunting (*cf.* Waibel and Lee 1990). The Fluent and Entropic systems evolved as research systems, and therefore emphasize convenient tools for creating limited-vocabulary CSR systems. Neither advertises an out-of-the-box large-vocabulary CSR system. Building a true, large-vocabulary continuous dictation system with their tools, while theoretically possible, is clearly beyond the scope of this project-- consider as evidence the fact that Lernout & Hauspie/Kurzweil have still not released such a product despite a $40 million investment from Microsoft last year.

In order to distinguish their products from small-vocabulary CSR systems, IBM and Dragon have taken to calling their systems "Continuous Dictation" (CD) systems. While research systems, like Fluent's and Entropic's, may introduce subtle improvements in basic technology, IBM and Dragon's CD systems are clearly the only speech recognition systems that meet the requirements of this project. The CD systems that we reviewed have the same range and single-speaker limitation as discrete dictation, but they can recognize large-vocabulary *connected* speech with better than 95% accuracy. The process of training these systems requires each new speaker to read a text aloud so a speaker model can be built. It takes about half an hour to record and build a speaker model, but the training can, in theory, be conducted on an instructional text as part of the instructional unit.

For purposes of speech recognition under the A97-099 ARI solicitation, CD systems are effectively required, and only the IBM and Dragon systems are currently acceptable. We nevertheless conducted a brief review of the remaining three systems on their text-to-speech capacity and for the possibility that they will release competitive CD systems within the time frame of A97-099. The SBIR specifies Windows95 and Windows NT as its delivery platforms. Although they have historically been developed for UNIX systems, Entropic and Fluent systems were evaluated because they have recently released Windows systems.

Entropic. Entropic has released *grapHvite for Windows.* We inquired for quotes on both UNIX and Windows versions of this system, comprised of HTK, Entropic's HMM modeling system and a graphical CSR-grammar authoring system. Entropic replied that they were delighted that we were inquiring about UNIX systems, because "that was their bread-and-butter". We replied that our first priority was the Windows system. After several more inquiries, Entropic has still failed to supply a quote or licensing information for *grapHvite for Windows.* We conclude that their commitment to Windows is thin, and that their support policies for Windows are poor. Although Entropic has been a leading name in university-based speech research, they appear now to be at least a year behind IBM and Dragon in the Windows CD market, and we are not confident they can place a competitive product into this market within the time limits of A97-099.

Fluent. Fluent CSR development has been running on Unix (and according to their technical contact it does not run in real time). They just recently released a Windows beta version of their *CSLUC Toolkit* (20 March 1998). Fluent recommended downloading a web-based version of the beta, but that version would not run across our proxy server. We then, according to Fluent's instructions, attempted to download the full (51M) beta-version, but their web download link was and has remained disabled. While OGI-UCSC has been a well-respected name in university-based speech research, we are not confident that Fluent will be able to place a competitive product into the Windows CD market during the time limits of this project.

IBM and Dragon. Of the two CD systems, *Naturally Speaking* narrowly bested *ViaVoice* on accuracy. In tests with three speakers, two male and one female, both Dragon and IBM consistently scored above 90%, and as speakers became accustomed and more "cooperative", scores ran as high as 96% (*ViaVoice*) and 98% (*Naturally Speaking*) on familiar vocabulary. *Naturally Speaking* was also slightly easier to use than *ViaVoice*. For example, with *Naturally Speaking* one could simply say, "scratch that", and *Naturally Speaking* would suggest a near homonym, or let the user rephrase what he had just spoken. Although it was also impressively accurate and seemed to keep up as well with dictation, *ViaVoice* annoyingly waited for long phrases to be spoken before it displayed its dictation. When errors did occur, this made it difficult to stop and correct them, and unlike *Naturally Speaking*, one could not correct them by voice.

However there are compensating considerations. First, *ViaVoice* placed its output directly into whatever application was currently accepting character input. The fact that *ViaVoice* was running in the background to do this may be what accounted for its slow screen output. (We did not have time to evaluate Dragon's *Word97* plug-in for *Naturally Speaking*. It is probable that when *Naturally Speaking* runs in the background it will exhibit the same slow screen output as we saw with *ViaVoice.*) Such background processing should be less of a factor as clock rates increase and more CPU cycles become available.

Second, IBM describes *ViaVoice* as a "speaker-independent" system. They claim that individual speaker training (IBM calls it "enrollment") is unnecessary for satisfactory performance. This claim is suspect—in our tests we needed enrollment to obtain satisfactory accuracy—but this attempted speaker independence does make *ViaVoice* more authorable than the *Naturally Speaking*. With *Naturally Speaking*, every speaker must individually add domain-specific vocabulary. With *ViaVoice* it is possible to add domain-specific vocabulary only once, speaker-independently. This may entail somewhat less recognition accuracy downstream, but the overall design philosophy is much more consonant with that of an authorable intelligent tutor.

Lernout & Hauspie. L&H is pursuing the Windows market much more aggressively. Last year, it acquired Kurzweil Speech Systems, and Microsoft has acquired a major ownership position within L&H

with a recent $40 million investment. Lernout & Hauspie has, in turn, purchased other firms, including, just last week, Apptek, a McLean-based machine translation company. At this writing, L&H's discrete dictation (one-word-at-a-time) technology is not acceptable for our project, but L&H could release CD technology as early as 2Q 1998.

Text-To-Speech Synthesis

Although phoneme-based speech synthesizers first appeared over 20 years ago (Klatt, 1976), text-to-speech technology has developed more slowly than speech recognition. Most text-to-speech engines can render individual words successfully. However, as soon as the engine speaks a sentence, it is easy to identify the voice as synthesized because it lacks human *prosody*—that is, the inflection, accent, and timing of speech. Every synthesizer had its anomalies, and none stood our test as better over all the others in terms of prosody. So, our evaluation centered on platform and interface compatibility and integration with animated talking heads.

Entropic. The Entropic text-to-speech system (TrueTalk, from AT&T) only runs under UNIX and was not evaluated further.

Fluent. Fluent licenses the University of Edinburgh *Festival* system. The *Festival* system does an acceptable job of synthesis, but it has only one male American English voice (a former graduate student of Loritz' from Georgetown). It does implement a wide range of low-level controls, but these are not SAPI-compliant. *Festival* is controlled by Scheme, but while this is a close cousin to Common Lisp, which will be used in our dialog system, it still complicates the design by the addition of one additional high-level interface language.

The following three products support the Microsoft SAPI-- Lernout & Hauspie, Dragon, and IBM.

Lernout & Hauspie. L&H's support is inferred from Microsoft's substantial investment interest in L&H and from the fact that the L&H TTS engine is embedded in the latest release of the Microsoft SAPI software developers' kit. For this evaluation we looked at the Lernout & Hauspie TTS system, *Willow Talk*, released in mid-1997. Several variants appear to have been released since then as part of Microsoft's agent technology and in the aforementioned MS-SAPI SDK. In *Willow Talk* and informal evaluation of these variants, however, the L&H synthesis was generally the least natural-sounding.

Dragon. We could only infer Dragon's support for the MS-SAPI from the fact that they request SAPI experience in several of their web site's technical job postings. An effort to directly ascertain their support by telephone calls to technical support yielded no help at all. We were referred to their developer's network, whose technical web pages are nearly a year out of date. We did, however, find an application form to join the Dragon beta test program. Two weeks after application, having heard nothing, we applied again. In all, it took Dragon 6 weeks to reply to our application for a Developer's Kit. This experience bodes ill for future Dragon technical support and we rate Dragon's technical support as poor.

Dragon's failure to reply to our technical TTS queries may simply reflect the fact that Dragon did not develop its own TTS technology, but instead has licensed it from Elan Technologies. This, however, leads to another shortcoming in the Dragon TTS package -- it supplies only one voice and that voice is a male voice speaking British English. Elan itself does not supply low-level SAPI-compliant drivers.

IBM. IBM's *ViaVoice* package supplies the strongest TTS engine in the group. Off-the-shelf, it supplies 8 pre-configured voices, as well as an easy authoring interface for creating a wide range of

variable additional voices. New voices can be created by a well-defined API to modify the head size (vocal tract length and cross-sectional area), fundamental frequency (bass/tenor), and timbre variables like breathiness. Almost all of these can be configured as slider bars to make voices fully authorable along dimensions such as, young/old, bass-coloratura, and hoarse/mellifluous. Below this level of IBM "SMAPI" authorability, ViaVoice's TTS software developer's kit further supports MS-SAPI, the Microsoft Speech API. The tools and documentation are excellent.

Talking Head

IBM. IBM's Developer Kit also includes *Virtual Voices*, its interface between MS-SAPI and the Eloquent TTS system. *Virtual Voices* has 8 talking-head "actors" (man, woman, child, and 5 caricature actors) with six "preset" expressions (happy, thoughtful, sad, neutral, surprised, asleep). These *Virtual Voices* are implemented as split-face bitmaps, such that top-frame eye-expressions (the "preset" expressions) can be controlled independently of bottom-frame lip expressions. The result is more caricature than natural, but it can be easily improved and extended to a wide range of highly authorable talking heads.

Dragon and Lernout & Hauspie. Dragon and L&H presumably could support a talking head akin to the *ViaVoice* design through MS-SAPI. There is no evidence Dragon has done so yet. Lernout & Hauspie has been integrated with animated Microsoft agents, but these do not yet appear to have been interfaced to TTSMOUTH, the SAPI talking head interface, to the level of *ViaVoice*. A viable alternative could be a graphically controllable face (one not especially configured for talking) such as *Smirk* ($299) or *Kinetix 3D Studio Max* ($3500).

Fluent (*Baldi* from UCSC). Fluent technologies offers *Baldi*, a "talking head" integrated with *Festival*, as part of their package. Baldi is technically far more complex, but the result is still far short of natural, and it cannot be nearly so easily authored. *Baldi* runs well on its development platform, a Silicon Graphics machine and was recently ported to a Windows NT environment. Additional work is needed to port the texture-mapping capabilities from the SGI platform and to reduce initial speaking latencies.

Special Hardware

All the PC systems require is a standard, Wintel-compliant sound card. We tested the three systems on both a SoundBlaster 32 ($79) and a SoundBlaster 16 clone card ($29). We could find no consistent differences in performance. Subtle differences may have existed, but in our tests they were *far* less important than the single factor of ambient noise. We conclude the speech systems need no more than a $29 sound card, which we expect will be standard equipment on PC systems when we deliver the product.

The greatest challenge in Talking Head technology is not so much lip-synching as getting the head to do things such as, look quizzical on questions, jut its jaw on contrastively stressed pronouns, and raise an eyebrow on counterfactual statements. Before this can be done, the head must have a 'brain' in it. It must have some Natural Language Understanding. We take that to be the real challenge that could be tackled in some future project. Our current objective is engaging, or at least entertaining caricatures, not photo-realism. So long as photo-realism is not an objective, no special hardware should be necessary to support Talking Heads in this project.

Conclusions

At this time, IBM's *ViaVoice* system is the best choice for our SAPI-compliant development platform.

Adopting the SAPI standard will leave a door open in the event that Dragon or Lernout & Hauspie should eclipse *ViaVoice*, but for the present neither can exceed *ViaVoice's* many strengths. Dragon's slight edge in accuracy is more than counterbalanced by *ViaVoice's* speaker-independent design philosophy, thus better supporting multiple users and authoring. In addition, IBM has excellent developer support tools and low-level TTS controls with their integrated talking heads.

Although *ViaVoice* demonstrates considerable progress in the field of speech recognition, it should be borne in mind that at 95% accuracy, speech recognition systems will still make an *air ore* in every other sentence. An *air ore* is what a CSR system often types when you dictate "error". Our dialog technology is uniquely capable of fixing most typos, spelling, and grammatical errors, and our dialog system will be capable of resolving some of the learner's semantic errors. Industry estimates (PCWeek, 3/4/98) are that it will be at least five years until a speech recognition engine can achieve 99% accuracy and that until then most users will prefer a keyboard interface.

PRODUCT COMPARISION TABLE

Speech Input

	ViaVoice IBM	*Naturally Speaking* Dragon	*VoicePad* Lernout & Hauspie	*CSLUC Toolkit* Fluent	*grapHvite* Entropic
Speech Recog	Continuous Dictation	Continuous Dictation	Discrete	Continuous Speech recog	Continuous Speech recog
Vocab size Total/Active	260K/22-64K	230K/30K	100K		
Accuracy	96/100	98/100			
Speaker-independent	yes	Not for domain specific vocab			
Allows multiple users	yes	yes			
Windows	yes	yes	yes	yes	maybe
Ease of use	90	95			
Ease of setup	90	95	95	0	
Support	Very good	Poor		Good	Poor
Est Cost	$100/ workstation	$100/ workstation	$50/ workstation	$50K/ project	>$10K

Text-to-Speech, Talking Head

	Eloquent IBM	*Elan* Dragon	*Willow Talk* Lernout/ Kurzweil	*Festival* Fluent	*TrueTalk* From AT&T In Entropic Pkg
Voices	8, variable	1 British male	2	1 American male	
Talking Head(s)	8 Virtual Voices Actors	No head but could support one	Integrated with MS agents	Baldi	
Technical Interface	Excellent, MS-SAPI TTSMOUTH	Unknown, MS-SAPI TTSMOUTH	But not interfaced to TTSMOUTH	Tcl/Scheme	Only runs in Unix -- not evaluated

Table 1. Product comparison table for spoken interface and talking head components.

Overall System Control and Delivery

The BB1 blackboard architecture, developed by Dr. Barbara Hayes-Roth at Stanford University's Knowledge Systems Laboratories, will be used as the integration and control tool. In general, BB1 supports complex knowledge-based applications that need to be able to plan their own actions. Each major component (e.g., ITS, conversational agent scenarios, speech recognition) is a module[2] that can be invoked by the blackboard. When modules are missing the system will still run, albeit with lesser capabilities (e.g., users can opt to omit speech recognition). We will build an open architecture that will allow us to incorporate later refinements and new modules.

The tutor will be implemented in a client-server fashion to allow both standalone and web-based delivery. For the web-based delivery, the speech recognition, synthesis and talking head modules will be installed and run from the individual's workstation. For the dialog and tutoring systems, we will allow user to choose either downloads from the net or else run from a central server. We don't anticipate any performance problems running from a central server, as long as usage is controlled to one trainee interacting with the system at a time. We could add multi-threading to accommodate multiple simultaneous users, but that is beyond the scope of the current effort.

The interface will be implemented in Java and available from any Java-enabled browser such as Netscape or Internet Explorer. The non-interface portion of the tutor will be implemented in Common Lisp. Communication will between the client and server will be through HTTP and the Common Lisp HTTP Server that runs on both UNIX and Windows NT platforms. Common Lisp is an ideal delivery language as it is platform-independent and it facilitates the kind of symbolic processing and reasoning needed for both the dialog system and the dynamic planner. The current dialog system is implemented in Scheme, a dialect of Lisp, which is readily portable to Common Lisp. The dynamic planner is based on the BB1 blackboard system (BB1 Version 3.2) and is implemented in Common Lisp and CLOS (Common Lisp Object System). The diagram below shows a sketch of the proposed setup.

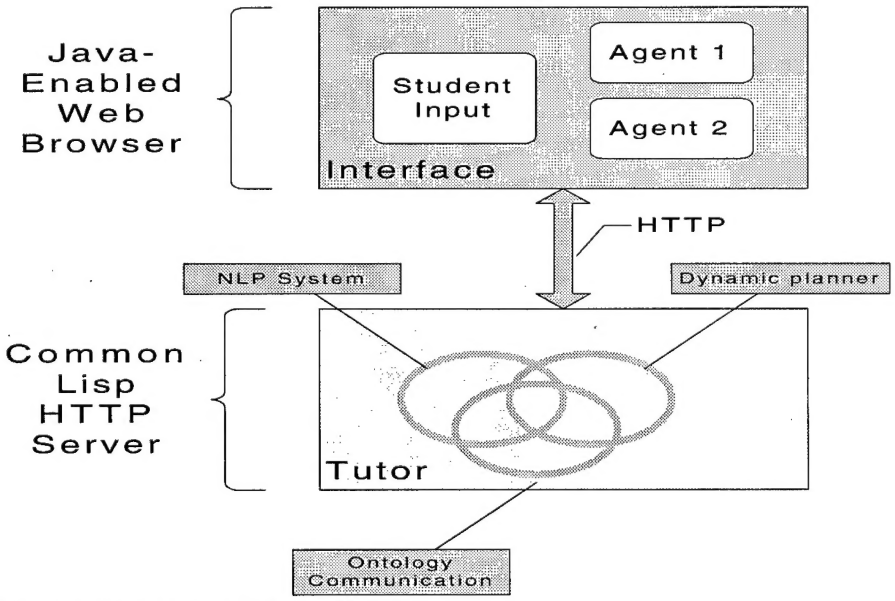


Figure 6. Overall system control and delivery.

[2] These modules may be further decomposed into sets of knowledge sources.

REFERENCES

Alshawi, H. (Ed.) (1992). *The core language engine.* Cambridge MA: MIT Press.

Brown, J.S., Burton, R.R., and de Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II, and III. In Sleeman, D. & Brown, J.S (Eds), *Intelligent tutoring systems.* Academic Press.

Carroll, J.B., Davies, P. & Richman, B. (1971). *The American heritage word frequency book.* Boston: Houghton Mifflin.

Clancey, W. J. (1987). *Knowledge-based tutoring—the GUIDON program.* The MIT Press.

DeSmedt, W.H. (1995). Herr Kommissar: An ICALL conversation simulator for intermediate German," in Holland, V. M., Kaplan, J.D., Sams, M.R. (Eds.), *Intelligent language tutors: theory shaping technology.* Erlbaum, 153-174.

Evett, M., Hendler, J. & Spector. L. (1990). *PARKA: Parallel knowledge representation on the connection machine*, UMIACS TR 90-22.

Grossberg, S. (1986). The adaptive self-organization of serial order in behavior: speech, language, and motor control. In Schwab, E.C. & Nusbaum, H.C. (Eds.), *Pattern recognition by humans and machines.* Orlando: Academic Press.

Hayes-Roth, B., Garvey, A., Johnson, M.V., & Hewett, M. (1987). A modular and layered environment for reasoning about action, *Technical Report KSL* 86-38, KSL, Stanford.

Jensen, K., Heidorn, G, & Richardson, S. (1993). *Natural language processing: the PLNLP approach.* New York: Kluwer.

Klatt, D. (1976). Structure of a phonological rule component for a synthesis-by-rule program. *IEEE Trans. Acoustics, Speech, and Signal Processing* ASSP-24, 391-8.

Koskenniemi, K. (1983). Two-level model for morphological analysis. *Proc. IJCAI* , 683-5.

Lenat, Douglas B. & Guha, R.V. (1990). *Building large knowledge-based systems*, Addison Wesley.

Loritz, D. (1993). Generalized transition network parsing for language study: The GPARS system for English, Russian, Japanese and Chinese. *Calico Journal 10*, 1: 5-22.

Loritz, D. (*in press). Rethinking cognition: the evolution of brain and language.* New York: Oxford University Press.

Marcus, M. (1980). *A theory of syntactic recognition for natural language.* Cambridge, MA: MIT Press.

McDonald, D. (1983). Natural language as a computational problem: an introduction. In Brady, M. & Berwick, R. (Eds.). *Computational models of discourse.* MIT Press. Cambridge, MA.

Murray, W.R. (1990). A Blackboard-based Dynamic Instructional Planner. In *Proceedings of the Eighth National Conference on Artificial Intelligence.* AAAI Press / MIT Press, pp. 434 - 441.

Psotka, J. L., Massey, D., & Mutter , S.A. (1988). *Intelligent tutoring systems: lessons learned.* Hillsdale, NJ: Erlbaum.

Rabiner, L.R. & Juang, B.H. (1993). *Fundamentals of speech recognition.* Englewood Cliffs, NJ: Prentice-Hall.

Reigeluth, C.M. (Ed.) (1987). *Instructional theories in action: lessons illustrating selected theories and models.*

Sato, P.T. (1988). A common parsing scheme for left- and right-branching languages. *Computational Linguistics 14*:1, 20-30.

Schafer, R.W. & Rabiner, L.R. (1975). Digital representation of speech. *Proc. IEEE 63,* 4:662-667.

Sowa, P. (Ed.) (1983). *Conceptual structures : Information processing in mind and machine.*

Waibel, A. & Lee, K.F. (Eds). (1990). *Readings in speech recognition.* San Mateo, CA: Morgan Kaufmann.

Williams, K (in press). An automated aid for the conduct of a detailed cognitive task analysis for modeling human computer interaction performance. In *Cognitive task analysis: theory and practice.* Lawrence Earlbaum Assoc.

Woolf, B.P. (1984). *Context-dependent planning in a machine tutor.* Doctoral dissertation, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts.

Woods, W.A. (1980). Cascaded ATN grammars, *AJCL 6*:1, 1-12.